ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seconda Facoltà di Ingegneria con sede a Cesena
Corso di Laurea in Ingegneria Informatica

APPLICAZIONI INFORMATICHE DELLA SWARM INTELLIGENCE

Elaborato in
FONDAMENTI DI INFORMATICA B

Tesi di Laurea di
LORENZO PONTELLINI

Relatore
Prof. ANDREA ROLI

Anno Accademico 2011 / 2012
II Sessione di Laurea

PAROLE CHIAVE

Divisione del lavoro

Soglia fissa

Soglia variabile

Stimolo

Adattamento

Auto organizzazione

Ai miei genitori che hanno sempre creduto in me

Indice

Introduzione

Di esempi di divisione del lavoro per aumentare l'efficienza di una fabbrica, di una città o di uno stato ne è piena la storia, ma tutto questo che collegamento trova con l'informatica? L'informatica, per quanto riguarda lo sviluppo di nuovi e sempre più efficienti algoritmi, prende spunto dal mondo animale, infatti studi svolti su insetti hanno portato alla luce una struttura sociale molto più complessa di quello che si può pensare. In realtà, anche all'interno di una colonia di insetti vi è una gerarchia di divisione dei compiti che mira alla sopravvivenza della specie cercando sempre una maggiore efficienza.

I componenti della colonia infatti collaborano non rispondendo ad un ordine diretto dettato da un individuo più importante degli altri che guida le sorti di tutti ma utilizzano metodi per comunicare discreti: ciò significa che le informazioni vengono assimilate da tutti e tutti sono in grado di trasmetterle. Queste vengono diffuse ad esempio modificando l'ambiente o variando i comportamenti dei singoli così che tutti riescano a reagire di conseguenza.

Il loro comportamento riesce ad essere robusto anche di fronte ai cambiamenti dell'ambiente circostante, riuscendo così ad adattarsi ad esso. Ciò che suscita maggiore interesse è il fatto che i singoli membri del gruppo non sembrano possedere ulteriori conoscenze per identificare, ad esempio, territori esterni rispetto a quello dove sono, non seguono un piano globale e addirittura possiedono una limitata capacità di lavoro; nonostante ciò riescono lo stesso a svolgere ricerche su territori vasti in cerca di cibo senza perdersi ritrovando sempre la strada per tornare al nido. Questa e altre operazioni sono molto superiori alle capacità del singolo quindi quello che suscita maggiore interesse è come riescano ad ottenere un comportamento di gruppo cosi malleabile partendo da tanti singoli.

Il meccanismo chiave del loro comportamento è l'auto-organizzazione.

I ricercatori risultano interessati a questo campo in quanto i possibili utilizzi sono svariati: dalla ricerca del percorso minimo di una strada, all'instradamento di pacchetti in rete, dalla ricerca di un oggetto allo studio di veicoli senza pilota.

Lo studio che si ha in informatica ha come obiettivo la scoperta di algoritmi sempre più efficienti per riuscire, con componenti semplici, a svolgere compiti complessi, con il minore carico di lavoro possibile. Le applicazioni di tale studio trovano risultati anche nel campo del controllo adattativo di robot.

Si vogliono confrontare tramite questo studio le osservazioni più importanti riguardati queste caratteristiche rese note dalla scienza e applicarle ai campi sopra citati per dimostrare l'effettivo valore e affidabilità che si guadagnano andando a utilizzare degli algoritmi che rispecchiano le stesse caratteristiche che si possono notare nel regno animale.

La metodologia di interesse usata come caso di studio è quella del recupero di oggetti. Esistono numerose soluzioni a questo problema che possono trovare uso in molte realtà utili all'uomo. Ne verranno presentate e confrontate due all'interno di questo elaborato, studiando le caratteristiche positive e negative di entrambe. Questi due approcci sono chiamati a soglia fissa e a soglia variabile. Entrambe sono tipologie di adattamento che prendono spunto dal comportamento che hanno le colonie di formiche quando si muovono alla ricerca di cibo. Si è deciso di analizzare queste due metodologie partendo da una panoramica generale di come cooperano gli insetti per arrivare al risultato finale, per poi introdurre nello specifico le caratteristiche di entrambe analizzando per ognuna i risultati ottenuti tramite grafici, e confrontandoli tra di loro.

Tra le due quella sulla quale si vuole porre l'attenzione è la seconda infatti si analizzano due approcci, uno per uno studio teorico ed uno per uno studio più pratico. Tramite il primo si scopre un algoritmo di base su come muta il comportamento delle formiche, altro non è che l'adattamento, questo viene tradotto in termini informatici e applicati ad un gruppo di robot i quali hanno il compito di ritrovare, in una apposita area dove si svolgono gli esperimenti, delle prede.

Per quanto riguarda i robot viene utilizzata una tecnologia chiamata lego mindstorm, questa risulta essere la prediletta in quanto sfrutta da una parte la facilità di creazione di robot, dall'altra la semplice reperibi-

lità dei componenti. Per rimanere sempre in sintonia con la realtà i robot non verranno dotati di grandi prestazioni, in quanto devono assomigliare in tutto e per tutto al singolo individuo di uno sciame di formiche quindi devono semplicemente essere in grado di muoversi, afferrare un oggetto e ritornare al nido. Gli individui non dovranno comunicare tra di loro, per lo meno non in maniera diretta, questo avverrà modificando l'ambiente che occupano (come per le formiche dove si ha il rilascio di una sostanza che gli altri componenti sono in grado di captare e in base alla concentrazione prendere la decisione corretta).

Si vorrà fare notare il contribuito dato dall'algoritmo di controllo in entrambi i casi e vedere che questo provocherà nei robot una certa divisione in categorie più o meno capaci nel portare a termine il compito di ricerca. Questo, nella simulazione, può avere un collegamento con il numero di successi ed insuccessi durante il recupero di una preda, mentre per quanto riguarda i robot reali, esso rispecchia la diversità di costruzione che ha effetto sul tipo di lavoro svolto. Il tutto per avere un riscontro pratico nella realtà dove esiste sempre una diversità biologica tra gli individui della stessa specie.

Per quanto riguarda i risultati ottenuti, alcune caratteristiche (come ad esempio il tempo di sperimentazione, il numero di individui nell'area di studio...) sono state adattate a ciò che si cerca di dimostrare per rendere anche più agevoli gli esperimenti.

La prima parte dell'elaborato porta una descrizione generale delle caratteristiche esibite dalle colonie di formiche nelle operazioni di ricerca e recupero del cibo e del perché si è scelto di analizzare questi insetti come caso di studio. Oltre a questo si presentano problemi che da sempre ricoprono un ruolo importante nell'informatica, cercando di dare una soluzione.

Nel secondo capitolo viene introdotta una definizione di divisione del lavoro con l'analisi di una prima metodologia di risoluzione, quella a soglia fissa, viene data una descrizione dell'esperimento collegata ad essa, ed un primo approccio alla specializzazione.

Nel terzo capito si analizza la metodologia a soglia variabile, con una descrizione accurata su come vengono condotti gli esperimenti, sia per quanto riguarda la parte di simulazione sia per quella di pratica. Per quest'ultima si portano delle caratteristiche della tecnologia lego mind-

storm indicandone le proprietà. Vi sarà poi una parte in cui si descrive come vengono condotti gli esperimenti con una definizione appropriata di efficienza adattata a questa tecnologia.

Per concludere in ogni capitolo verranno mostrati tramite grafici i risultati ottenuti per facilitare il confronto tra le due metodologie proposte, analizzando per ognuna i pro e i contro riscontrati.

Capitolo 1

Swarm Intelligence

Con il termine **Swarm Intelligence** tradotto letteralmente come sciame intelligente, si usa indicare un insieme di comportamenti che portano alla realizzazione di un'azione complessa tramite una intelligenza complessiva di più individui, come per l'appunto nel caso di sciami di formiche. Con esso si usa indicare gli studi svolti su sistemi auto-organizzanti. Secondo la definizione di Gerardo Beni e Watt la swarm intelligence può essere definita come: *Proprietà di un sistema in cui il comportamento collettivo di agenti (non sofisticati) che interagiscono localmente con l'ambiente, produce l'emergere di pattern funzionali globali nel sistema".*
Quando si parla di swarm intelligence, nel mondo animale si fa riferimento spesso al termine Ant Foraging il quale si riferisce alla ricerca di cibo da parte di colonie di formiche.
Questo aspetto ha interessato, ed interessa tuttora, numerosi studiosi sia naturalisti che informatici in quanto ogni insetto in una colonia sembra essere in possesso delle informazioni necessarie allo svolgimento del proprio compito, così come il gruppo nel suo insieme sembra essere altamente coordinato.

1.1 Introduzione allo studio

Lo studio fatto sui metodi di ricerca del cibo da parte delle formiche ha contribuito alla creazione di algoritmi utilizzati nella direzione del traffico di rete in sistemi di telecomunicazioni saturi, oltre a questo gli algoritmi sono stati perfezionati ed adattati per la ricerca di oggetti da parte di robot.
Un pioniere di questo studio è stato Denenubourg il quale ha mostrato che le formiche per riuscire a trovare il percorso più breve tra il nido ed

una fonte di cibo lasciano delle sostanze[1] nell'ambiente modificandolo e informando così gli altri componenti del gruppo.
Uno degli esperimenti condotto da Denenubourg è stato quello di costruire un ponte tra il nido delle formiche e la fonte di cibo con due sole strade una lunga il doppio dell'altra, si è visto che in poco tempo le formiche muovendosi inizialmente in maniera casuale alla ricerca di cibo nell'area interessata, hanno selezionato la strada più corta.

1.2 Applicazioni della swarm intelligence

Le applicazioni della swarm intelligence ricoprono svariati campi, di seguito vengono spiegati alcuni esempi di applicazione dove la soluzione ottimale è stata trovata utilizzando queste tecniche viste nel mondo animale. Seppur questi problemi sembrano astratti, nella realtà in cui l'uomo vive si presentano svariate volte seppure sotto forme differenti.

1.2.1 Problema del commesso viaggiatore

La soluzione a questo tipo di problema richiede di trovare il percorso più breve che passa attraverso un dato numero di città solamente una volta. La soluzione richiede un numero di calcoli che cresce con il numero di città da attraversare.

Il professor Marco Dorigo ha mostrato che questo tipo di problema può essere risolto trovando il percorso ottimale utilizzando le formiche artificiali le quali vengono programmate in modo che la concentrazione di feromone da loro rilasciata vari con la distanza percorsa.

1.2.2 Gestione delle reti

Una chiamata telefonica tra A e B deve passare in generale tra un numero di nodi intermedi o stazioni di commutazione e richiede dei meccanismi che selezionino la successiva stazione intermedia attraverso la quale passare. Il tutto naturalmente evitando le aree congestionate per minimizzare il ritardo. Nel caso in cui le condizioni cambino drasticamente bisogna prevedere anche degli apparecchi di backup per rigenerare il segnale.
Per fare fronte a tutte queste condizioni sono stati inventati delle tecniche di instradamento dei pacchetti prendendo spunto dalla ricerca del

[1] La sostanza rilasciata dalle formiche è il feromone

cibo delle formiche basate su un tipo di feromone virtuale per instradare il percorso verso aree non congestionate.

1.2.3 Controllo di un gruppo di robot

Una prima applicazione che si sta studiando è quella del trasporto cooperativo. Utilizzando questo approccio è possibile creare robot relativamente semplici ed economici in grado, ad esempio, di assemblarsi quando le condizioni lo richiedono e creare un robot complesso capace di fare fronte alle caratteristiche per le quali il singolo non è adatto.
Questo esempio può spiegare come partendo dal singolo componente, il cui potenziale per svolgere operazioni è basso, si riesce ad arrivare ad un gruppo organizzato o meglio auto-organizzato che risulta in grado di portare a termine compiti complessi.
Un gruppo cosi composto offre alternative per la progettazione di sistemi che hanno tradizionalmente richiesto un controllo centralizzato e di pre-programmazione estesa, infatti vanta caratteristiche di autonomia e autosufficienza basandosi su interazioni dirette e indirette tra gli agenti individuali, tale situazione porta a dei sistemi che possono adattarsi rapidamente anche a condizioni che possono variare.

Capitolo 2

Divisione del lavoro

Un fattore chiave che contribuisce al notevole successo ecologico degli insetti è la loro organizzazione sociale, e in particolare la divisione del lavoro. Si è dimostrato che la divisione dei compiti tra i membri di una colonia, in modo che gli individui tendano a diventare specializzati in certi ruoli, migliora l'efficienza della colonia. Questo sia perché i componenti sviluppano caratteristiche specifiche e competenze attraverso la pratica, sia perché riduce il dispendio di tempo e di energia che i componenti del gruppo consumano nei movimenti tra le diverse posizioni.

In questo elaborato verranno presentati due approcci che hanno come argomento la divisione del lavoro: il primo sarà il metodo a soglia fissa mentre il secondo prevede una soglia variabile, entrambi verranno utilizzati per effettuare un confronto finale tra le due metodologie.

2.1 Modello a soglia fissa

Il modello assume che i lavoratori siano in grado di valutare le esigenze attraverso particolari stimoli che determinano l'esecuzione delle attività, e che le soglie di risposta non variano nel tempo. Quando gli individui, caratterizzati da soglie di risposta basse rispetto a stimoli relativi a una determinata attività, sono isolati (chiamati ad esempio minori), la domanda relativa aumenta, così come l'intensità dello stimolo, fino a raggiungere una soglia. L'incremento della intensità dello stimolo oltre la soglia ha l'effetto di stimolare questi individui nello svolgere il compito. Questo tipo di modello prevede un algoritmo che simula la divisione del lavoro basato su una soglia fissa.

2.1.1 Introduzione al modello

La prima importante domanda alla quale porre una risposta è di capire come la flessibilità risulta implementata al livello del singolo individuo.

Questo tipo di problema può essere diviso in due parti:

- Come i componenti del gruppo trovino e raccolgano le informazioni loro necessarie sia nel caso in cui debbano iniziare un nuovo lavoro sia quando ne riprendono uno interrotto;

- In che modo essi decidono come comportarsi una volta che possiedono le informazioni necessarie.

Per quanto riguarda il primo punto si assume che ad ogni lavoro sia associato uno stimolo o un insieme di stimoli[1]. L'intensità di questi stimoli che l'animale percepisce contiene abbastanza informazioni, dunque quando l'individuo si trova in contatto con una fonte di stimoli valuta la richiesta associata ad essi.

Vengono fatte ora due assunzioni: la prima è che in un dato intervallo di tempo l'individuo riesca a percepire tutti gli stimoli, la seconda risulta essere che tutti gli stimoli abbiano la stessa probabilità di essere captati.

Nel proseguimento del documento si vuole porre l'attenzione sul secondo punto precedentemente citato.

L'algoritmo che viene utilizzato è il **FTM**[2] in questo modello si assume che gli individui siano caratterizzati da una soglia di risposta fissa agli stimoli presenti.
Dal punto di vista di questo algoritmo si generano diversi gruppi di individui caratterizzati da proprietà comuni a tutti i componenti. Questi gruppi, risponderanno al nome di *caste*, mentre il lavoro che dovranno compiere verrà indicato con *task*.

Quando gli individui concludono un task vengono isolati (questo perché possiedono una soglia di risposta bassa agli stimoli presenti) la relativa domanda incrementa e così fa anche l'intensità degli stimoli fino a che la soglia di risposta caratteristica dei rimanenti individui, che inizialmente

[1] Uno stimolo viene inteso come un segnale forte e affidabile correlato a specifiche richieste

[2] *Fixed Threshold Model*: appunto modello a soglia fissa

non sono specializzati nel lavoro, non sia più alta. L'incremento dell'intensità degli stimoli oltre la soglia ha l'effetto di stimolare gli individui nella conclusione dei task.

2.1.2 Osservazioni sperimentali legate al modello

Verrà ora spiegato il significato della **soglia di risposta**. Per fare questo occorre introdurre determinati parametri:

- **S**: intensità dello stimolo associata al particolare task.
  Può essere un numero, oppure una concentrazione ormonale, o qualsiasi parametro che può essere recepito dall'individuo;

- Θ: è il limite di risposta espressa in unità di intensità di stimoli.
  E' una variabile interna che determina la tendenza di un individuo a rispondere agli stimoli;

Una famiglia di funzioni di risposta che può essere parametrizzata con una soglia che soddisfa i requisiti prima citati è data da:

$$T_\Theta = \frac{s^n}{s^n + \Theta_i^n} \tag{2.1}$$

2.1.3 Modello a soglia fissa con un task e due caste

In questa prima parte si assume che sia eseguito solo un task, il quale, risulta essere associato ad uno stimolo.
La domanda relativa allo stimolo incrementa se il task non è soddisfatto, questa maggiorazione si assume che sia fissa per ogni unità di tempo. Si ipotizza, per questo esperimento, che gli individui siano divisi in due **caste** caratterizzate dalla propria soglia di risposta alla domanda.

Per come è stato introdotto il problema, la scala temporale sulla quale si lavora è assai ristretta e grazie a questo, si può considerare la soglia di intervento fissa.
La divisione in caste può riflettere la reale diversità degli elementi della colonia sia in termini di età dei componenti del gruppo, sia in termini di tipo di lavoro affidato al singolo, oppure semplicemente per esprimere in maniera fedele la biodiversità degli insetti.

In quanto segue si considera:

- **X**: lo stato in cui si trova l'individuo. Se uguale ad 1 allora significa che l'individuo sta svolgendo un task, nel caso invece che sia uguale a 0 significa che è in uno stato di attesa e inattività;

- Θ_i: è la soglia di risposta della i-esima casta.

Un individuo nello stato di inattività passa allo stato di lavoro con probabilità P_i per unità di tempo:

$$P_i(X = 0 \longrightarrow X = 1) = \frac{s^2}{s^2 + \Theta_i^2} \tag{2.2}$$

Nella figura 2.1.3 viene mostrata la curva di risposta che corrisponde alla formula 2.2 per il gruppo dei maggiori e minori dove si assume che la distinzione tra i due gruppi sia in riferimento alle caratteristiche fisiologiche.
Potrebbe inoltre rappresentare diversi attributi comportamentali, o semplicemente una distinzione di età.

Il gruppo dei maggiori spesso è caratterizzato da una minore probabilità di diventare attivo rispetto minori.

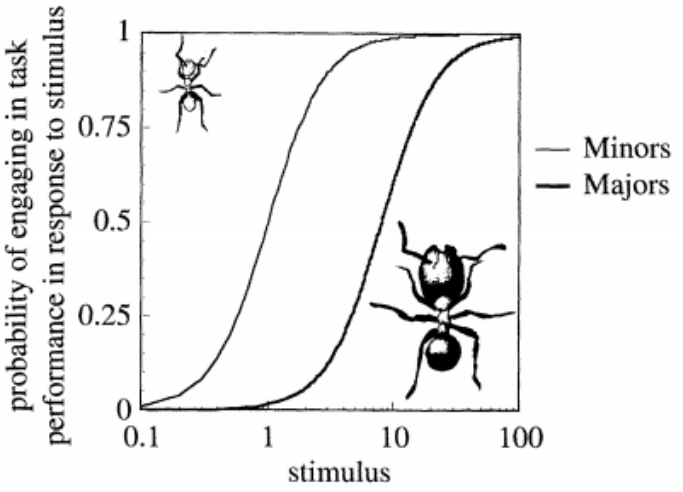


Figura 2.1: Curva di risposta con diverse soglie per i maggiori e minori

Un individuo nello stato attivo, appartenete alla casta i-esima (i = 1,2) lascia il task e diventa individuo inattivo con probabilità p per unità di tempo.
Si considera identica per entrambe le caste:

$$P_i(X = 1 \rightarrow X = 0) = p \tag{2.3}$$

Il risultato p può essere trovato sperimentalmente dato che si considera $\frac{1}{p}$ come il tempo medio speso da un individuo nella risoluzione di un task prima che smetta di occuparsi di esso. Si assume che p sia fisso e indipendente dagli stimoli.

Gli individui, occupati nella risoluzione di un task, spendono un tempo corrispondente a $\frac{1}{p}$ anche se il loro lavoro risulta completo e non più necessario per portare a termine il task.

In qualche caso si è verificato che gli individui dopo aver lavorato per un tempo di $\frac{1}{p}$ lasciano il task ma ripassano immediatamente allo stato di operativo se lo stimolo è ancora persistente.

Le variazioni dell'intensità dello stimolo possono essere provocate da:

- la risoluzione del task che riduce l'intensità degli stimoli;
- il naturale incremento della domande che è indipendente dal fatto che il task venga completato o no.

L'equazione che risulta per l'evoluzione dell'intensità degli stimoli $\mathbf{s}$ è quindi:

$$s(t+1) = s(t) + \delta - \frac{\alpha}{N}(N_1 + N_2) \tag{2.4}$$

dove:

- N_i è il numero di individui appartenenti alla casta i-esima che sta eseguendo il task;
- N è il numero totale di individui potenzialmente attivi;
- δ è l'incremento in intensità degli stimoli per unità di tempo;
- α è un fattore di scala che misura l'efficienza delle performance del task.

Vengono fatte alcune ipotesi: (1)l'efficienza nella risoluzione del task è la stessa che si ha sia per gli individui appartenenti alla casta 1 che per quelli appartenenti alla 2; (2) Si assume che l'efficienza non vari significativamente. Quest'ultima condizione può essere dichiarata in quanto la scala dei tempi è ristretta e quindi gli esperimenti si concludono in un

tempo sufficientemente corto, mentre l'apprendimento e quindi l'adattamento può prendere parte solo su lunghi momenti temporali.

Un'ulteriore assunzione che viene fatta è che nell'equazione (2.4) la quantità del lavoro svolto dagli individui attivi è scalata per N per riflettere l'idea che la domanda è una funzione incrementale del fattore N. Si considera inoltre una funzione lineare.
Questo termine rispecchia l'idea che il gruppo richiede una fattore di scala lineare con la dimensione della colonia, data la presenza di questo fattore di scala, ci si aspetta che i risultati siano indipendenti dalle dimensioni della colonia.

La figura 2.2 mostra un grafico sulla comparazione dei risultati di una simulazione con N = 10 e N = 100 individui. La media del tempo speso per i maggiori è espressa in (individui x step di tempo) in funzione della frazione di maggiori nella colonia[3]. Come si vede dal grafico, il numero di attori per individuo subisce una rapido cambiamento quando la frazione dei maggiori supera alcuni valori critici (dalla figura 2.2 si nota che questo è 0,4). Tutti gli altri parametri sono stati puntualizzati in modo da prevenire che gli stimoli diventino più piccoli delle soglie.

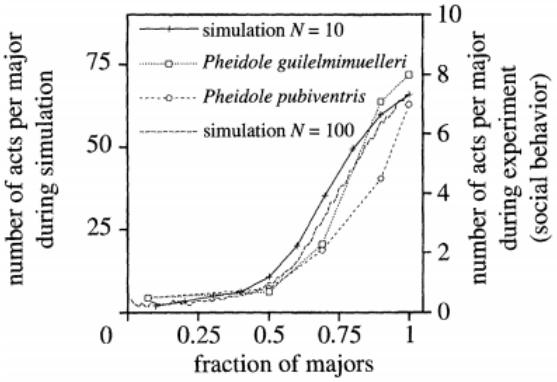


Figura 2.2: Grafico di comparazione tra i risultati

La figura 2.3 mostra che i maggiori possono rispondere ad una richiesta diventando più evoluti nella risoluzione di un task quando c'è un improvviso incremento nel ritmo di aumento della domanda associata a quel task. Molti minori sono già coinvolti nella risoluzione di un task prima di questo cambiamento così che il coinvolgimento dei maggiori è

[3]Questi sono i risultati di un esperimento condotto da *Wilson*

richiesto per mantenere abbastanza alta la domanda.

Questo modello può facilmente giustificare, tramite osservazioni sperimentali, che il gruppo dei maggiori torna al task originale una volta che il minori siano tornati alla colonia.

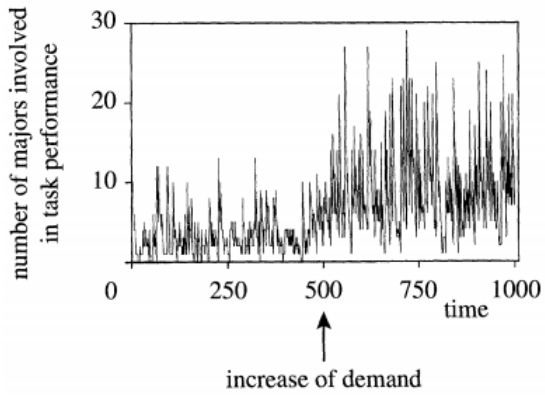


Figura 2.3: Evoluzione nel tempo del numero dei maggiori durante la risoluzione del task

Come ultima immagine si considera la 2.4, la quale mostra come le curve vengono modificate quando il rapporto $z = \frac{\Theta_1^2}{\Theta_2^2}$ incrementa: la transizione diventa più brusca, e il punto iniziale tende a diminuire.
Quando z è vicino ad 1, la proporzione dei maggiori impegnati nella risoluzione di un task parte da un valore più grande rispetto a quando z è maggiore.

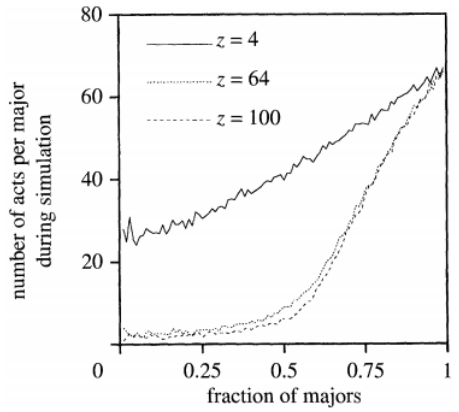


Figura 2.4: Individui nei maggiori in funzione del rapporto di z

2.1.4 Modello a soglia fissa con due task e due caste

Si assume in questo caso che ci siano due task distinti che debbano essere portati a termine da parte di due individui appartenenti a due caste distinte.
E' possibile generalizzare il modello appena descritto ma si possono avere due casi:

- il gruppo dei minori è specializzato in entrambi i task ma le soglie di intervento sono differenti(le quali risultano più basse rispetto al gruppo dei maggiori);
- entrambe le caste sono specializzate, quella dei maggiori in un task, quella dei minori nell'altro.

Vengono considerate valide tutte le relazioni che fino a qui sono state discusse, le quali con opportuni accorgimenti potranno essere applicate ad entrambi i due casi che di seguito verranno descritti.

Minori hanno soglia di intervento più bassa in entrambi i task

Quando i minori possiedono soglia di risposta più bassa per entrambi i task, le curve praticamente sono simili a quelle ottenute nell'esempio generale.
Nella figura 2.5 si vede il numero di attori per due task durante la simulazione come una funzione della frazione dei maggiori del gruppo.
Questi risultati vengono comparati con quelli ottenuti da *Wilson* che ha misurato la stessa quantità per entrambi i task.

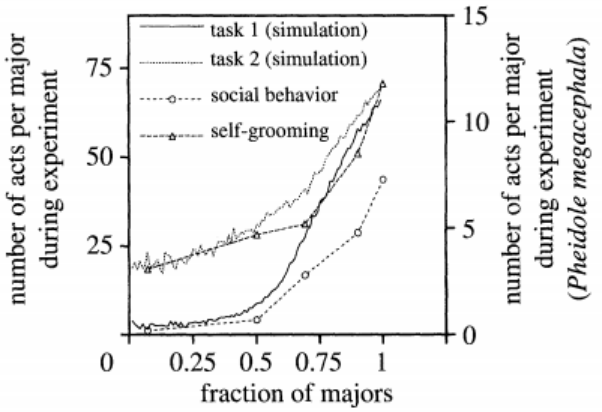


Figura 2.5: Grafico per la comparazione di risultati con gli esperimenti condotti da *Wilson*

Una prima curva mostra un rapido cambiamento a bassi valori della frazione dei maggiori, mentre la seconda mostra un cambiamento più omogeneo ad un valore più alto.
Il risultato può essere compreso mostrando che la proporzione di maggiori che sono impegnati nella risoluzione di un task varia con il rapporto di soglia **z**: se il task 1 corrisponde ad un valore di **z** più grande rispetto a quello del task numero 2, allora la curva associata al task 2 subisce un cambiamento più brusco rispetto a quella del task numero 2.

Entrambe le caste sono specializzate in un task

Quando una casta risulta specializzata in uno dei due task, e l'altra casta nell'altro task, si osserva un comportamento più flessibile per entrambi i gruppi: i maggiori possono sostituire i minori e viceversa.

Un individuo appartenente ai minori non può sempre essere spinto a lavorare su un task risolto usualmente dai maggiori, mentre è sempre vero il contrario cioè che gli appartenenti al gruppo dei maggiori possono essere spinti alla risoluzione di un task appartenente al gruppo dei minori.

Questo ragionamento però non può essere applicato alle caste fisiche, ma può essere applicato senza problemi alle caste meno rigide.
Si può modellare quanto esposto, assumendo che i minori abbiano una soglia di risposta agli stimoli associati ai task dei maggiori molto alta (virtualmente infinita), in linea di principio qualunque soglia susciterà la risoluzione del task una volta che l'intensità dello stimolo diventerà troppo alta.

2.1.5 Possibile approccio alla specializzazione

Attraverso l'uso della soglia fissa è possibile introdurre una prima definizione di specializzazione, facendo uso però della seguente ipotesi: si considera che sia possibile variare la percezione dello stimolo in funzione delle performance di risoluzione del task.

Accettando questo si può parlare di specializzazione spaziale dove per gli individui di un certo tipo che svolgono un task è più probabile imbattersi in uno stimolo piuttosto che altri individui.
Ad esempio come succede in per alcune specie di formiche dove il gruppo di esse intente alla raccolta di cibo è più propenso a difendere il nido in caso di allarme anche se il comportamento difensivo non è il loro scopo

primario.
La spiegazione a questo fatto è che le formiche che ricercano cibo sono in contatto più facilmente di altre ai pericoli dell'ambiente.

Più in generale l'esecuzione di un compito può promuovere o ridurre l'esposizione diretta a stimoli associati con altre attività, o può promuovere o ridurre incontri con altri individui che stiano svolgendo altre attività. Questi incontri possono anche essere regolati e sfruttati come controllo verso gli altri componenti del gruppo per valutare il numero di lavoratori che stanno svolgendo un task. In assenza di regolazione, come nel caso in cui si verifichino troppi incontri con gli individui che stanno svolgendo un task, si inibisce la risoluzione del task da parte degli altri individui, il numero di task da svolgere dovrebbe essere fisso indipendentemente dalla grandezza della colonia.

Relazione tra specializzazione e spazio - tempo

Un supporto alla tesi sopra citata è che l'esposizione a stimoli dipende dal tipo di task che si sta svolgendo, infatti esperimenti mostrano che in alcuni casi i task vengono organizzati anche a livello di **spazio**.
La relazioni spaziali tra le attività di solito sono pensate per essere un fattore di efficienza, infatti se i lavoratori tendono ad eseguire una serie di compiti spazialmente localizzati, i cammini medi per muoversi da uno all'altro posto in cui si eseguono i task sono ridotte al minimo e le attività sono più facilmente localizzate.

Se si tiene conto di una ulteriore variabile: il **tempo** abbiamo quanto segue. In questi casi, individui più giovani tendono a essere situati all'interno del nido, vicino al suo centro, mentre gli individui più anziani sono più probabilmente adibiti ad eseguire le attività al di fuori del nido, come la difesa o la ricerca di cibo.

La spiegazione che si può trovare per tutto ciò è che la relazione età-posizione gioca un ruolo importante anche per la sopravvivenza della colonia.
Infatti i lavoratori più giovani vengono esposti meno a pericoli, quindi tendono a rimanere nel nido dove possono deporre le uova, mentre i lavoratori più anziani avendo meno probabilità di essere in grado di riprodurre, vengono spinti ad eseguire attività più pericolose al di fuori del nido.

La figura seguente mette in relazione la risoluzione dei task con l'età dell'individuo.

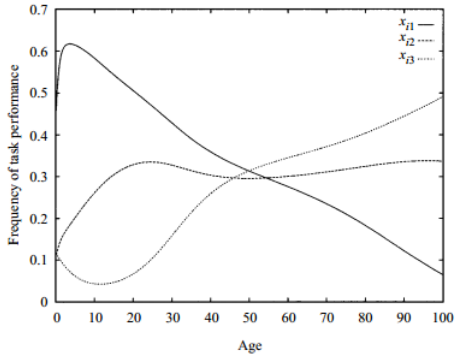


Figura 2.6: Risoluzione del task in funzione dell'età

2.1.6 Discussione risultati ottenuti

Da quanto detto fino ad ora attraverso il modello **FTM** si è in grado di spiegare, almeno parzialmente, la flessibilità del comportamento osservato in una colonia di formiche. Gli stessi tipi di esperimenti potrebbero essere svolti su specie diverse di formiche per conoscere e capire come varia la soglia di risposta in relazione al cambiamento della specie di colonia, questo risulta sicuramente interessante ma certamente più difficoltoso che affrontare lo studio su di una singola specie.

Gli studi sul metodo **FTM** vengono applicati, come scritto precedentemente, ai casi che si verificano in un sufficientemente breve lasso di tempo mostrando che la probabilità della reversione dal task B al task A è una funzione decrescente, dipendente dal tempo speso durante il task B.

Capitolo 3

Modello a soglia variabile

Come discusso nei capitoli passati si è visto che la natura è piena di esempi in cui animali possono cooperare efficientemente anche se in grandi gruppi. Le formiche ne sono un esempio: esse infatti possono recuperare collettivamente anche una grossa preda oppure trovare il percorso più breve verso una fonte di cibo.

Tale comportamento risulta essere anche robusto di fronte alle caratteristiche dell'ambiente o a repentini cambi di condizioni. Si vuole ricordare inoltre, quanto detto prima cioè che nella maggior parte dei casi al singolo individuo del gruppo sembra mancare una mappa interna dell'ambiente oppure un piano globale al quale rispondere. Il meccanismo chiave sul quale si basano è l'**auto-organizzazione**.

Di qui in avanti si studieranno i comportamenti di un gruppo di robot in relazione a quanto appreso dalle abitudini delle formiche. Lo studio verrà condotto con due tipi di robot uno realmente costruito mentre l'altro sarà solamente simulato, con essi verranno condotti degli esperimenti per vedere come cambiano le prestazioni considerando e non una forma rudimentale di apprendimento.

L'applicazione che è stata decisa come test è quella di un ideale recupero di una preda da parte di un gruppo di robot: il test prevede la ricerca all'interno di una area apposta della preda, la cattura, e il trasporto fino al nido.

3.1 Prima definizione di rendimento

La prima cosa che occorre definire è un indice di valutazione delle prestazioni del robot. Per definire questo indice occorre valutare quali sono i fattori che lo influenzano in maniera positiva e negativa.
Da una parte il recupero può essere visto come un *guadagno* per il gruppo, mentre la ricerca e il recupero possono essere viste come *costi* dato che costituiscono un pericolo, questo perché sono azioni che espongono i robot alle insidie dell'ambiente, e al fatto che spendono energia per muoversi.

Proventi e costi dipendono entrambi da X il numero di robot che stanno esplorando l'ambiente, considerando che i guadagni saturano quando X diventa troppo alto (i robot non possono raccogliere più prede del loro numero), e che i costi possono aumentare senza limiti si ha una prima definizione di rendimento così:

$$\eta = \frac{guadagni}{costi} \tag{3.1}$$

Esistono tre modalità per aumentare l'efficienza:

- fornendo migliori strumenti ai robot in modo che impieghino meno tempo nella ricerca;
- inserendo una comunicazione tra i robot;
- utilizzando un numero ottimale di robot.

La strada che verrà perseguita sarà l'ultima. Esiste, infatti, un numero che massimizza il rendimento, ci si riferisce a questo anche in base ad un meccanismo già discusso che è l'adattamento. Questo può essere fatto ponendo la seguente condizione: le caratteristiche dell'ambiente sono fissate.

3.1.1 La ricerca di prede nelle colonie di formiche

Si riassumono di seguito, in quanto già discusse nei capitoli precedenti, le caratteristiche della ricerca di cibo all'interno delle colonie di formiche:

- le formiche esplorano l'ambiente in maniera casuale fino a quando una di loro non trova una preda;

- quando la trova cerca di tirarla al nido, nel caso non riesca da sola o cerca di tagliarla oppure attua un reclutamento a corto raggio spargendo nell'ambiente una sostanza che gli altri componenti del gruppo riescono a percepire e decodificare come richiesta;

- dopo il recupero la formica rientra dove ha trovato la preda.

3.2 Metodi

Si descrive in questo capitolo l'approccio di studio utilizzato. Vengono utilizzati per gli esperimenti due tipi di robot con differenti caratteristiche:

- **robot reali**: vengono costruiti dei prototipi reali di robot facendo uso della tecnologia Lego Mindstorms^{TM}

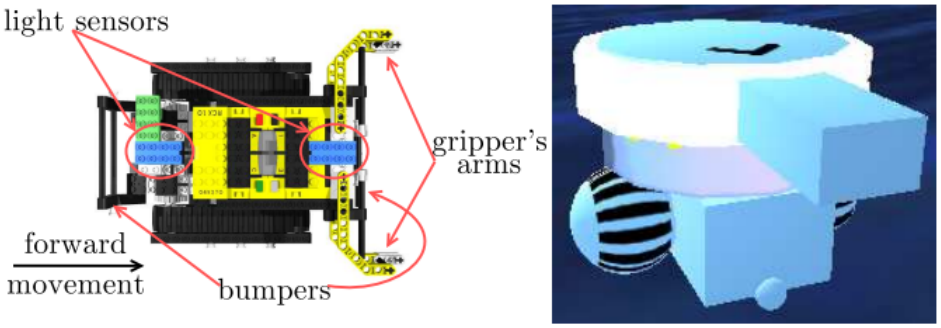
- **robot simulati**: sono robot autonomi creati nel progetto SWARM-BOTS[1].

La ragione che ha portato ad utilizzare due tipi di robot è la seguente: seguendo questo approccio si riuscirà a trarre delle conclusioni più generali e meno dipendenti da condizioni sperimentali.

Per validare il modello teorico si useranno i robot reali, questi risultano migliori per questo compito in quanto si evita che ci sia un convalida errata a causa di parametri che non sono stati scelti in modo corretto.
Una volta che si è validato il modello si può passare alla fase di simulazione la quale permette di risparmiare tempo dato che con il metodo della simulazione si riescono ad ottenere risultati accelerando la fase di analisi.

Tramite i robot reali, si controlla il modello teorico e si definisce un percorso per una ulteriore analisi, con la simulazione si incrociano i dati ottenuti con quelli del simulatore e si può proseguire verso una analisi più approfondita.
Una prima immagine per spiegare le differenze tra le due tipologie di robot è la figura 3.1:

[1]Sito internet di riferimento: http://www.swarm-bots.org



(a) The *MindS-bot*. (b) The *s-bot*.

Figura 3.1: Rappresentazione di un *MindS-bot* a sinistra, mentre a destra un *s-bot*

3.2.1 Robot reali

I robot reali che verranno chiamati *MindS-bot* (Figura 3.1(a)), sono stati costruiti con la tecnologia Lego Mindstorms^{TM}.

La CPU che usano è una Hitachi H8300-HMS 1 MH, con 32kb sia per il sistema operativo sia per il programma utente. Nella figura vi è la rappresentazione dall'alto, si possono notare due sensori di luce per il riconoscimento delle prede e la localizzazione del nido. Si è dotato inoltre il robot di due paraurti uno davanti uno dietro per aggirare gli ostacoli. La CPU controlla tre motori due per il movimento e uno per le percorso. Numeri casuali, utilizzati per alcune applicazioni, vengono generati da BrickOS[2].

Task cui il robot è sottoposto

In generale un robot deve possedere le seguenti caratteristiche:

Modello dinamico di formazione/aggiornamento il robot deve essere in grado auto-organizzarsi in un numero di configurazioni differenti scegliendo la più adatta al task che sta svolgendo e all'ambiente nel quale si trova per fare fronte alle modifiche che può subire il suo habitat.

Movimento in terreni accidentati la colonia di robot deve essere in grado di muoversi in terreni accidentati utilizzando le informazioni dei singoli *s-bot*.

[2]Il sistema operativo utilizzato per la programmazione dei robot

Tirare/spingere oggetti la colonia di robot deve essere in grado di tirare po spingere degli oggetti che sono troppo pesanti per il singolo robot. In caso di bisogno il singolo robot deve essere in grado di richiedere aiuto e auto-assemblarsi per portare a termine il compito raccogliendo le informazioni sull'ambiente e comportarsi di conseguenza.

Suddivisione dei task solo alcuni robot di tutta la colonia devono occuparsi del trasporto dell'oggetto mentre altri devono comunque continuare la ricerca.

Lego Minstorm^{TM}

Lego Mindstorm^{TM} è la linea di prodotti Lego utilizzato per costruire il Mind-bot. All'interno della scatola vi sono i pezzi base: mattoni squadrati, assi, ingranaggi e differenti tipi di articolazioni, oltre a un po di componenti elettrici ed elettronici.

Come detto prima vi è il mattoncino con la CPU, poi sul lato anteriore, vi è un trasmettitore/ricevitore ad infrarossi per comunicare con un computer o con altre unità o per ricevere comandi da un telecomando. Un programma nella ROM si prende cura di download del sistema operativo da un computer mediante la porta ad infrarossi. Il sistema operativo esegue programmi utente e gestisce l'hardware ad interrupts. Un display LCD a cinque cifre risiede sulla parte superiore del mattone oltre a quattro tasti e slot per tre ingressi e tre dispositivi di uscita situati attorno. Le connessioni in uscita permettono di dare potenza agli attuatori, come i motori e le luci. Le connessioni in ingresso sono utilizzate per il campionamento e la digitalizzazione dei segnali con un convertitore analogico/digitale a 10 bit.

Per la realizzazione dell'attività di recupero preda, un Minds-bot deve essere forte abbastanza per tirare una preda. La forma circolare della s-bot è difficile da replicarla con i mattoncini squadrati di Lego.
Si è scelto di utilizzare una preda di colore nero in un ambiente bianco e una luce per segnalare la posizione del nido. La pinza deve essere in grado di trattenere una preda, due bracci, disposti simmetricamente rispetto al centro del robot e azionati dal motore stesso, formano la pinza.

Nella figura 3.2 sotto vi è la vista del robot completamente assemblato.

Figura 3.2: Vista rendering del Mind-bot

Software

Un apposito mattoncino esegue il programma di default nella memoria ROM alla prima accensione. Il programma attende tramite la porta ad infrarossi per un firmware, noto anche come sistema operativo, da scaricare. Da Lego viene fornito un firmware originale, che purtroppo è stato progettato solo per funzioni base. Il suo più grande svantaggio è che i programmi utente non possono contenere più di 32 variabili. Inoltre, i programmi sono compilati in un bytecode che è interpretato tramite il firmware, questo processo rallenta il tempo di esecuzione. Il software Lego non supporta altre caratteristiche che sono tipiche dei sistemi operativi, come ad esempio multitasking o il blocco delle risorse.
Ultimamente sono stati fatti sviluppi superando queste limitazioni riuscendo a scrivere programmi proprietari. In rete è possibile recuperare il software libero scritto in Perl o Java.

All'interno degli esperimenti svolti nello studio preso come riferimento[3] si è scelto per gli esperimenti di utilizzare *BrickOS*[4] il grande vantaggio di questo è che usa il codice proprietario del microprocessore anche per l'esecuzione di programmi. E' possibile scrivere programmi in qualsiasi linguaggio di programmazione e compilarli con un compilatore che supporta il codice corretto della macchina.
GCC[5] che è il compilatore più usato all'interno di distribuzioni Linux e

[3] *Division of Labour in a Gropup of Robots Inspiered by Ants' Foraging Beagaviour*
[4] http://brickos.sourceforge.net/
[5] http://gcc.gnu.org/

Unix, risulta essere già configurato per il tipo di processore usato.
BrickOS utilizza tra 14Kb e 18Kb di memoria, a seconda delle opzioni che l'utente desidera utilizzare. Il resto (18 - 14 Kb) è lasciato ai programmi utente. Inoltre, BrickOS implementa le seguenti caratteristiche:

- basato su priorità *preemptive multitasking*;
- processi di sincronizzazione basati sui semafori;
- CPU a risparmio energetico;
- controllo completo LCD per il debugging;
- ingresso per la gestione di tasti;
- un generatore di numeri casuale;
- è implementato un protocollo di comunicazione basato sulla comunicazione infrarossi che realizza: diffusione di messaggi indiretta, comunicazione stile UDP tramite indirizzo e porta.

I protocolli di comunicazione implementati in BrickOS non garantiscono che i destinatari ricevano i messaggi. Due robot (o un robot e un computer) devono posizionarsi in modo che le porte infrarossi siano allineate per comunicare e il sistema di controllo deve fare in modo che questa situazione sia possibile. Questa condizione non può essere garantita, mentre i robot si muovono. La funzione svolta da BrickOS è solo quella di inviare i pacchetti garantendo che, quando uno di questi viene ricevuto, l'immagine non sia corrotta.

3.2.2 Robot simulati

Il simulatore utilizzato per gli esperimenti è chiamato *swarmbots3d* ed è stato sviluppato dai membri del progetto SWARM-BOTS. E' un simulatore dinamico basato su Vortex[6], tramite esso è possibile simulare robot in grado di esplorare l'ambiente autonomamente e dà la possibilità di collegare tra loro più robot dinamicamente quando le condizioni ambientali lo richiedo.
Tra le caratteristiche principali di *swarmbots3d* vi è quella di poter simulare diverse tipologie di robot: si può andare da un semplice cilindro dotato di ruote e quindi in grado di muoversi, ad una riproduzione fedele

[6]http://cm-labs.com/

del robot reale. Ovviamente più dettagliata sarà la simulazione e maggiore sarà il tempo necessario richiesto per svolgerla.

Sono presenti dei sensori di luce simulati attorno al corpo principale del robot e una camera omnidirezionale, la quale ha una percezione limitata, il tutto utilizzato per localizzare la preda e il nido. Sensori ad infrarossi sono posizionati attorno a tutto il corpo per renderlo in grado di evitare gli ostacoli. La simulazione della pinza frontale è semplificata ma funzionale. Il parallelepipedo di fronte, come si vede nella figura 3.1(b), funge da calamita in grado di attaccare e staccare la preda dinamicamente creando un collegamento tra la preda e il robot. Numeri casuali sono generati da l'algoritmo *Mersenne Twister*[7]

[7]algoritmo implementato all'interno di *GNU Scientific library* http://www.gnu.org/software/gsl/

3.3 Metodi di controllo

In figura 3.3 si mostrano le transizioni della macchina a stati per il controllo dei robot reali e di quelli simulati.

Verrà data una definizione per ognuno di loro:

- **Search** Il robot cerca una preda. Serve per evitare la collisione con altri robot. Se una preda viene trovata, il Minds-bot afferra. Se è da troppo tempo alla ricerca di una preda senza averne trovata alcuna, si arrende;
- **Retrieve** Il robot ricerca il nido e tira la preda fino ad esso;
- **Deposit** Il robot lascia la preda nel nido e si gira rivolgendo la parte posteriore verso il nido;
- **Give up** Il robot sosta nel nido prima di riprendere la ricerca;

Le transizioni avvengono in base ad eventi che possono non dipendere o dipendere dal robot. Le etichette che sono poste ai bordi del grafico determinano le condizioni che devono essere verificate perché avvenga la transizione.

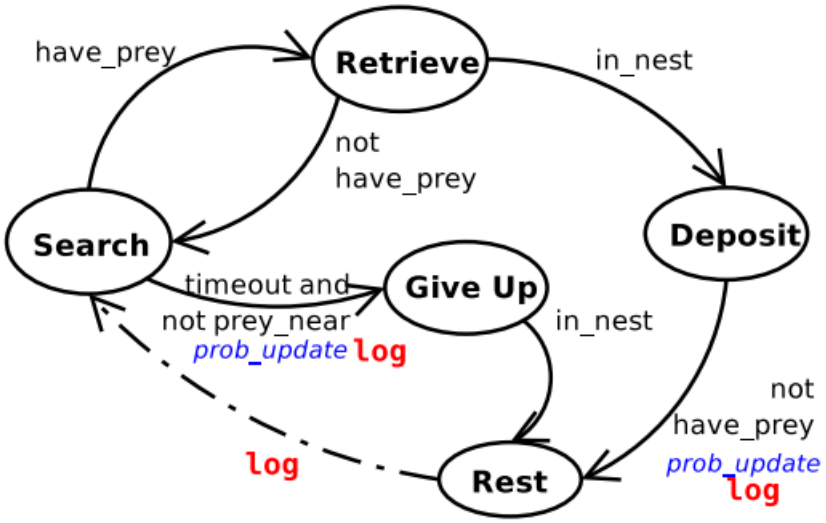


Figura 3.3: Stati della macchina astratta per la rappresentazione del controllo dei robot

La transizione tra **Rest** e **Search** è probabilistica. La probabilità viene aggiornata una volta che si verifica una transizione tra **Search** e

Give up nel caso di fallimento, o tra **Deposit** e **Rest** in caso di successo.

Il robot utilizza un certo numero di comportamenti, cioè sotto-procedure che reagiscono in base a ciò che viene captato dai sensori in ingresso. Ogni comportamento può essere eseguito solo se alcune condizioni di attivazione sono vere. Teoricamente si possono generare delle situazioni in cui i comandi che si dovrebbero applicare vanno in conflitto, per evitare questo occorre definire una gerarchia dei comandi.
Il programma di controllo sfrutta la funzione di multitasking di BrickOS. Ogni comportamento viene implementato come un processo autonomo che si occupa di leggere l'entrata e controlla i motori. Un *processo arbitro* controlla sia i cambiamenti di stato sia i comportamenti di attivazione. Se rileva che la condizione di attivazione di un processo con un più alto livello, in termini di gerarchia, è vero, il processo in esecuzione in quell'istante viene cessato e il nuovo si avvia.

Sia il processo arbitro che il controllo sul comportamento ripetono la loro esecuzione ogni 100 ms. BrickOS non fornisce caratteristiche real-time, pertanto, il periodo non è sempre preciso e dipende dallo scheduler del sistema operativo. Tuttavia, si considera che la precisione cosi raggiunta sia sufficiente per gli esperimenti qui esposti.

Infine, vi è un processo principale che controlla il processo arbitro. Il suo compito è solo attendere l'avvio e il comando di arresto da parte dell'utente per caricare i dati di login sul computer.

Nella tabella 3.4 viene mostrata la lista dei comportamenti che è possibile ritrovare in ogni stato, i comportamenti sono ordinati secondo priorità di attivazione: quello con priorità maggiore è il primo della lista.
Viene ora data una breve descrizione degli stati presenti nella tabella 3.4:

- **AvoidObstacle**: il robot cerca di evitare un ostacolo. Il suo primo controllo è sul paraurti frontale, attua quindi una rotazione per cercare di aggirarlo se lo ha colpito di fronte, altrimenti semplicemente si muove in direzione opposta aggirandolo nel caso la collisione sia avvenuta con la parte posteriore;
- **CloseGripper**: chiude la pinza frontale sulla preda;
- **OpenGripper**: apre la pinza frontale;
- **Explore**: è uno stato che fa partire l'algoritmo di movimento, tramite il quale il robot si sposta in maniera casuale in un'area dispo-

nibile, finché non trova una preda, nel caso non la trovi l'algoritmo si riavvia scegliendo un'altra direzione;

- **SearchNest**: il robot ricerca il nido tramite l'uso dei sensori di luminosità in quanto esso sarà rappresentato da un punto luminoso nell'area di interesse;
- **LeavePrey**: nel caso in cui il robot abbia catturato una preda si posiziona in maniera da essere orientato verso il centro del nido e rilascia la preda;
- **ExitFromNest**: questo comportamento non muove il robot ma tramite un algoritmo seleziona un numero casuale tra 0 e 1 ogni secondo, se il valore è maggiore della probabilità attuale di lasciare il nido allora segnala al processo arbitro che può passare allo stato di ricerca.

State	Behaviour	Conditions
Search	*CloseGripper*	`hit_prey`
	AvoidObstacle	`front_bumper` $\wedge\neg$`have_prey` $\vee$ `back_bumper`
	OpenGripper	`gripper_closed` $\wedge\neg$`prey_in_gripper`
	Explore	
Give Up	*AvoidObstacle*	`back_bumper`
	SearchNest	
Retrieve	*AvoidObstacle*	`back_bumper`
	SearchNest	
Deposit	*LeavePrey*	`have_prey`
	SearchNest	
Rest	*ExitFromNest*	

Figura 3.4: Tabella comportamento del robot in base alla verifica della condizione

Algorithm 1 Variable Delta Rule. P_l is the probability of leaving the nest.

```
initialisation:
successes ← 0
failures ← 0
P_l ← INITIAL VALUE

if prey retrieved then
  successes ← successes + 1
  failures ← 0
  P_l ← P_l + successes * Δ
  if P_l > P_max then
    P_l ← P_max
  end if
else if timeout then
  failures ← failures + 1
  successes ← 0
  P_l ← P_l - failures * Δ
  if P_l < P_min then
    P_l ← P_min
  end if
end if
```

Figura 3.5: Algoritmo per il controllo della probabilità di uscita dal nido del robots

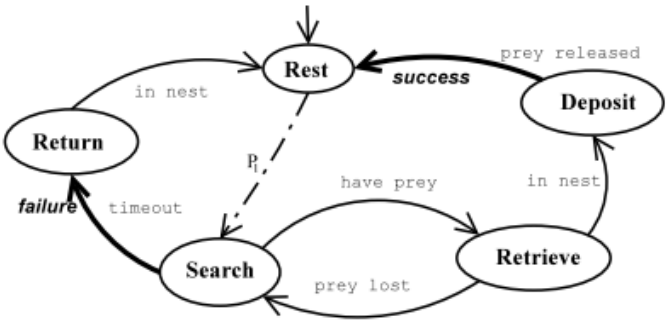
Nella figura 3.5 viene mostrato l'algoritmo di controllo per la variazione P_l cioè la probabilità con cui si lascia il nido.
Non ci si vuole soffermare troppo sull'algoritmo di figura 3.5 in quanto non verrà utilizzato quest'ultimo come algoritmo di controllo dato che tramite test preliminari si è dimostrato che le dinamiche di adattamento sarebbero state troppo lente.

La soluzione che si è trovata permette un adattamento più rapido, penalizzando però gli esperimenti con i robot reali in quanto hanno richiesto un tempo di 40 minuti ciascuno[8].

L'algoritmo originale quindi è stato modificato ottenendo quello in figura 3.6, viene inoltre modificato lo schema generale degli stati, i quali singolarmente mantengono sempre il loro significato quindi quanto detto fino ad ora rimane ancora valido.

[8]La durata degli esperimenti è generalmente in funzione non solo di questi fattori riguardanti l'adattamento ma anche da altri come ad esempio la velocità dei robot, e il loro metodo di ricerca

La transizione dalla posizione di **riposo** a quella di **ricerca** avviene ogni secondo con probabilità P_l, il cui valore viene aggiornato durante la transizione da **cerca** a **ritorno** che corrisponde ad un **successo** e da **deposita** a **riposo** che corrisponde ad un **fallimento**.
La probabilità di lasciare il nido P_l viene modificata in maniera simile a quanto descritto da Deneuborg: in questo algoritmo incrementa o diminuisce di una costante fissa Δ, nel caso in questione invece Δ viene moltiplicato per il numero di successi o fallimenti consecutivi.
P_l è delimitato da un valore nell'intervallo [P_{min} - P_{max}].



Algorithm 1 Adaptation rule of P_l, the probability to leave the nest. The variables *succ* and *fail* are the number of consecutive successes and failures.

$succ \leftarrow 0$; $fail \leftarrow 0$; $P_l \leftarrow P_{\text{init}}$;

if success **then**
  $succ \leftarrow succ + 1$
  $fail \leftarrow 0$
  $P_l \leftarrow min\{P_{\text{max}}, P_l + succ \cdot \Delta\}$
fi

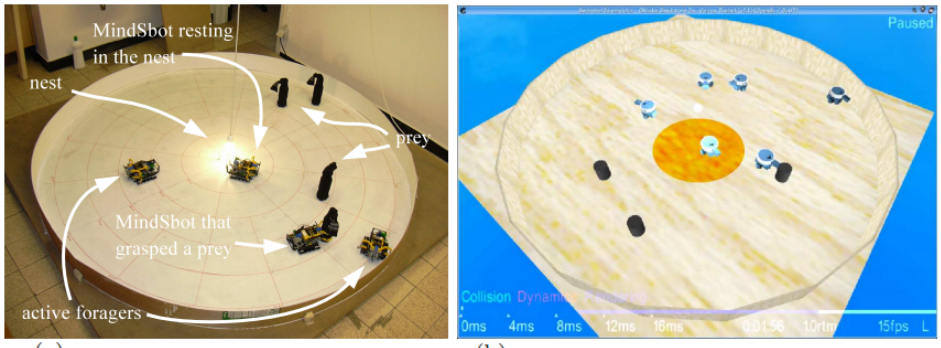
if failure **then**
  $succ \leftarrow 0$
  $fail \leftarrow fail + 1$
  $P_l \leftarrow max\{P_{\text{min}}, P_l - fail \cdot \Delta\}$
fi

Figura 3.6: Nuova macchina degli stati, e algoritmo modificato per l'aggiornamento della P_l

3.4 Set-up dell'esperimento

Per gli esperimenti è stata utilizzata un'arena circolare con un diametro di 2,40 metri (si veda la figura 3.7). Una lampadina viene posizionata al centro dell'arena e utilizzata per segnalare la posizione del nido. Pareti e pavimenti sono di colore bianco mentre le prede sono di colore nero.
Il timeout per la ricerca della preda per i **Mind-Sbot** è fissato a 228 secondi, mentre per gli **s-bot** è 71,2 secondi[9].

P_{min} è fissato a 0,0015 mentre P_{max} risulta essere 0,05 un ulteriore valore è P_{init} uguale a 0,033 valori che corrispondono ad una media di tempo trascorso nel nido rispettivamente di: 11 minuti, 20 secondi, 30 secondi.
Δ è fissato a 0,005.
Le prede appaiono in maniera casuale nell'ambiente, la probabilità con cui questo avviene ogni secondo è chiamato densità di preda e varia tra gli esperimenti.
Una nuova preda viene posizionata in maniera casuale all'interno dell'arena in una posizione compresa tra 0,5 metri e 1,1 metri dal centro. I valori scelti e utilizzati per gli esperimenti sono stati ponderati in modo da fare risultare più evidenti gli effetti dell'adattamento.



(a) (b)

Figura 3.7: Rappresentazione dell'arena per gli Mind-sbot (a) e dell'arena degli s-bot (b)

[9]questi tempi sono dei valori stimati necessari ad un robot per trovare una preda quando è da solo nell'arena, questi valori non dipendono solo dalla velocità dei robot ma anche dai sensori. l'**s-bot** riesce a trovare una preda grazie alla fotocamera omnidirezionale, mentre per quando riguarda **Mind-Sbot** la preda deve essere davanti a lui per essere localizzata

3.5 Indice di efficienza

Non è possibile definire il valore di η così come definito nel capitolo 1 (equazione 3.1), dato che per come sono stati precedentemente definiti i costi sono difficili da quantificare, infatti comprendono troppi fattori e alcuni dei quali sconosciuti. Per questo si è deciso di considerare costo il **tempo speso dal gruppo**[10]. Questo valore è la somma dei tempi spesi da tutti i robot alla ricerca o nel recupero della preda, cioè il tempo di servizio. Il tempo di servizio è direttamente collegato ai costi: più è elevato maggiore è la probabilità che qualche robot si perda o si danneggi. Cosi il concetto di efficienza è stato ridefinito come segue:

$$\nu = \frac{performance}{\sum_{robots} dutytime} \tag{3.2}$$

Dove *performance* è il numero di prede trovate.

3.6 Esperimenti e risultati

Vengono di seguito analizzati gli effetti dell'adattamento individuale in un gruppo di robot; le tematiche affrontate sulle quali ci si basa sono tre:

- efficienza di miglioramento;
- divisione del lavoro;
- selezione degli individui migliori.

3.6.1 Efficienza

Robot reali

Sono stati utilizzati dei gruppi di quattro *MindS-bot*, scelti tra un totale di sei. Dopo ogni esperimento i quattro robot sono stati sostituiti[11]. Ogni esperimento ha avuto una durata di 2400 secondi (40 minuti). Sono state generate dieci istanze con densità di prede fissate a 0,006 s^{-1}. ogni esperimento poi è stato ripetuto con lo stesso gruppo di controllo e composto dagli stessi robot con P_l fissata, sono state usate inoltre le

[10] Termine originale group duty time

[11] Questa scelta non è stata del tutto casuale ma dettata anche da altre cause ad esempio robot con la batteria scarica, oppure robot che risultavano danneggiati venivano scambiati con altri funzionanti

stesse istanze.

In figura 3.8 vi è il riassunto dei dati che sono stati elaborati, si mostra ν per entrambi i gruppi quelli controllati sia per quelli che usano l'adattamento. Come si vede dall'immagine quando i robot usano l'adattamento, in media nell'arena si possono trovare 2,57 robot attivi e 2,44 prede in un periodo di tempo tra i 1000 e 2400 secondi, mentre negli esperimenti dove i robot vengono controllati i robot attivi sono stati 3,63 e le prede 3,49.

Simulazione

Per quanto riguarda la simulazione sono stati utilizzati gruppi che variavano da 2 a 8 *s-bot* con incrementi di due unità. I gruppi, questa volta, sono stati testati con densità di preda variabile: (1) 0,005 s^{-1}, (2) 0,01 s^{-1}, (3) 0,04 s^{-1}. Sono state generate 50 istanze, sono state usate per tutte le combinazioni preda/dimensione gruppo.

Gli esperimenti hanno avuto una durata di 2400 secondi. Anche in questo caso gli esperimenti sono stati ripetuti utilizzando le stesse istanze di robot, ma con gruppi di controllo che non hanno utilizzato l'adattamento. I risultati anche questa volta sono stati raccolti nella figura 3.9.

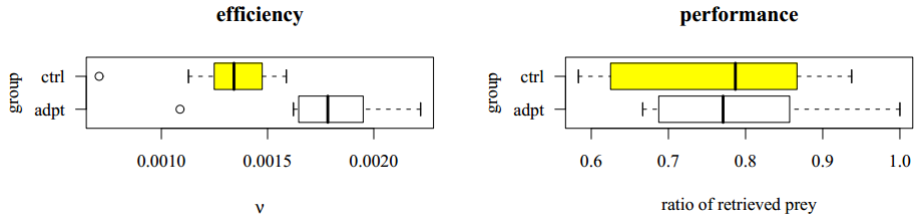


Figura 3.8: A sinistra: valori di ν quando si usa adattamento (sopra) e quando non lo si una (sotto) per i *MindS-bot*. A destra performance relative all'adattamento e non del gruppo di robot simulati

Discussione dei risultati

Il gruppo che ha usato adattamento è significativamente più efficiente sia nel caso dei robot reali che in quello delle simulazioni. Non sono presenti differenze statistiche nelle performance dei due gruppi di *MindS-bot* mentre nella simulazione il gruppo di controllo si comporta meglio, in quest'ultimo caso la differenza media di recupero prede è di 3,4 unità

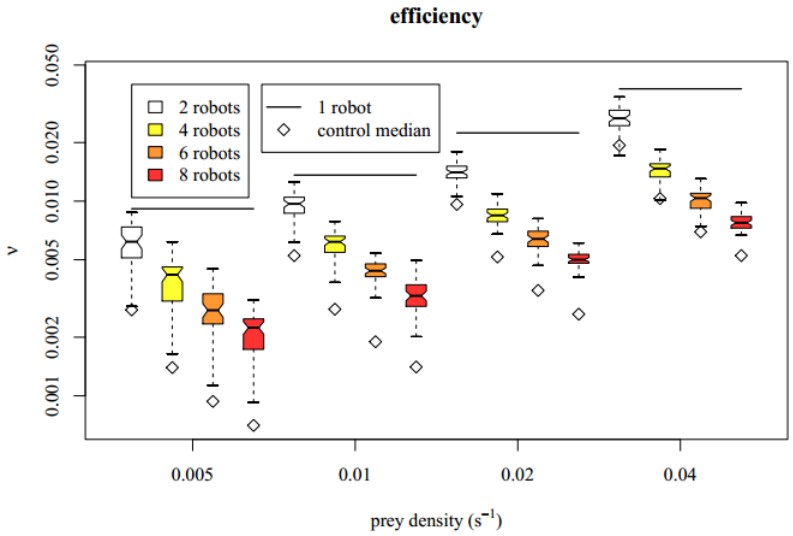


Figura 3.9: Effetti sull'efficienza della densità di prede e le dimensioni del gruppo

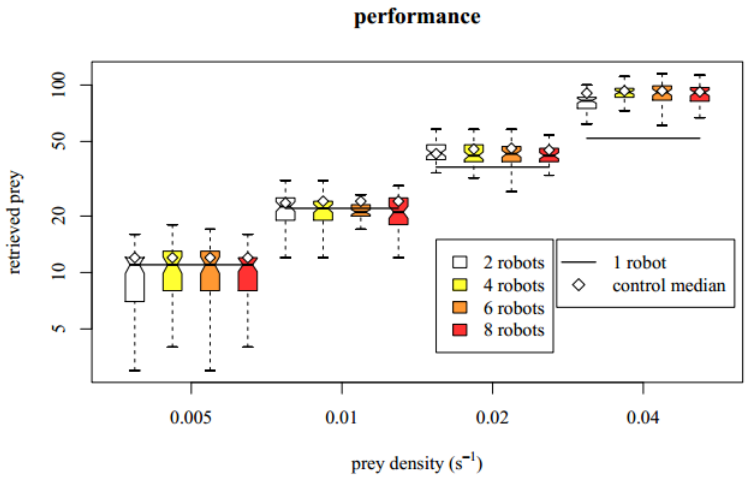


Figura 3.10: Prestazioni di differenti set-up in simulazione

trascurabile quindi rispetto al totale. La dimensione del gruppo sembra non influenzare l'andamento dello stesso. Una spiegazione a questo può essere data dal fatto che il numero relativo di prede recuperate, cioè il numero di prede recuperate diviso per il totale è vicino ad 1. Il gap tra il gruppo di adattamento e il gruppo di controllo tende a diminuire quando la densità di prede tende ad aumentare.

E' importante osservare che ν decrementa con la dimensione del gruppo anche quando si ha l'uso dell'adattamento. Questo è dovuto al fatto che l'algoritmo utilizzato non è efficiente come quello originale, oppure è legato al fatto che ν dipende dal *duty time*, e quest'ultimo dipende dalle dimensioni del gruppo.
Infatti tutti i robot spendono parte del tempo nella ricerca della preda, contribuendo ad aumentare il *duty time* finale del gruppo. Questo perché nessuno di loro può avere la P_l a zero, per le ipotesi fatte, dato che è compresa tra due valori maggiori di zero.

3.6.2 Divisione del lavoro

Si è mostrato che l'adattamento migliora l'efficienza del gruppo. La differenza di prestazioni sia in caso di simulazione sia nel caso reale, quando presenti non possono spiegare il miglioramento. Pertanto si deduce che l'adattamento riduce il *duty time* del gruppo. Ci sono due modi in cui il gruppo può raggiungere questo obbiettivo:

- tutti i robot terminano avendo lo stesso valore di P_l sebbene basso, questo significa che il numero di robot raccoglitori è ridotto;
- solo alcuni robot sono dei raccoglitori attivi con un alto valore di P_l, mentre gli altri possiedono un valore basso.

Ovviamente i robot con un valore alto di P_l potranno spendere un tempo maggiore nella ricerca rispetto agli altri, quindi in questo modo è possibile osservare divisione del lavoro.

In ogni istante di tempo t dall'inizio dell'esperimento, il valore di P_l di un robot è una variabile casuale che ha dei valori differenti per ogni robot e ogni esperimento.
Nel caso che il gruppo utilizzi la divisione del lavoro alla fine degli esperimenti la distribuzione di P_l possiede due picchi, altrimenti si avrà solamente un picco.

Robot Reali

Durante gli esperimenti sono stati annotati i valori di P_l nel tempo per ogni *MindS-bot* in modo da tracciarne la possibile distribuzione. I risultati dopo 2400 secondi sono stai riportati in figura 3.11 e chiaramente si vedono due picchi. La figura 3.12 mostra la distribuzione di P_l nel tempo.

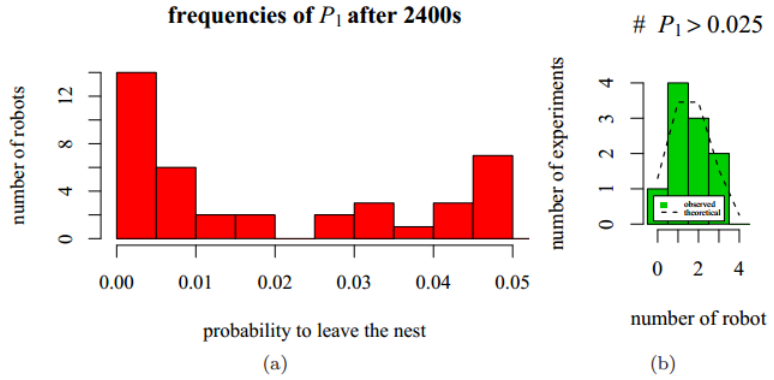


Figura 3.11: Frequenza di P_l osservata nei MindS-bot a 2400 secondi

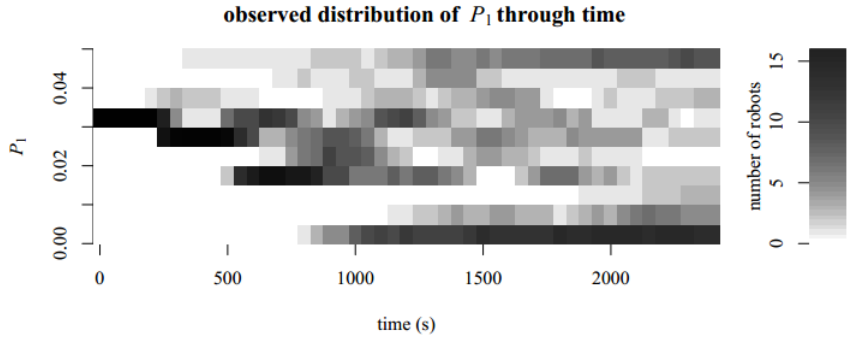


Figura 3.12: Distribuzione osservata nel tempo di P_l

Simulazione

Sono stati analizzati gli stessi dati utilizzati nella sezione riguardante l'efficienza, questa volta per analizzare gli effetti della dimensione del gruppo e di densità di prede sulla divisione del lavoro. Si è riportata in figura 3.6 (b) per ciascuna combinazione di densità di prede e dimensione del gruppo e si nota un divario più ambio tra i due picchi.

Si è scelto di classificare i robot in tre categorie: **foragers**, **loafers**, **undecided**:

- **foragers**: raccoglitori, sono quei s-bot la cui P_l è maggiore di 0,042;
- **loafers**: fannullone, sono quei robot in cui la P_l è minore di 0,007;
- **undecided**: indecisi, sono tutti gli altri robot, si noti che questa categoria presenta una gamma di P_l di cinque volte più ampia rispetto agli altri due.

In figura 3.13 si mostra la proporzione s-bot per ogni categoria al tempo di 2400 secondi, come si vede dai grafici vi è una forte divisione del lavoro. I robot tendono ad avere valori di P_l alti o bassi, raramente possiedono valori intermedi.

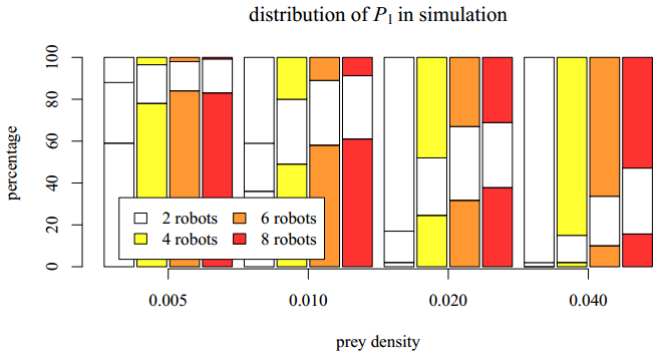


Figura 3.13: Distribuzione di P_l nel tempo

Discussione dei risultati

Sarebbe possibile obiettare che il picco sulla destra della figura 3.11 potrebbe essere il risultato di alcuni esperimenti in cui i *MindS-bot* è capitato di avere un alto valore di P_l. Per confutare subito questa tesi basta guardare il numero di *MindS-bot* con P_l maggiore di 0,025 al termine degli esperimenti e come questa risulta distribuita.
Dai dati raccolti si sa che il 40% della popolazione ha P_l maggiore di 0,025, pertanto ci si aspetta che il numero di *MindS-bot* in ciascun esperimento segua una distribuzione binomiale, controllando questo con i dati raccolti si vede che si assomiglia molto.

L'evoluzione della P_l nel tempo mostra che i *MindS-bot* con un valore più alto della stessa, appaiono nell'arena più tardi rispetto a quelli con un

valore basso. Tutti i *MindS-bot* iniziano con lo stesso valore si P_l, dopo del tempo alcuni per alcuni di loro il valore di P_l diminuisce in quanto non hanno successo, mentre per gli altri si alternano successi ad insuccessi. Il numero di *MindS-bot* nell'arena diminuisce, ci sono cioè meno concorrenti per coloro che sono riusciti a mantenere la loro P_l abbastanza alta, meno concorrenti significa un numero maggiore e più semplice di recuperi delle prede.

Come previsto il numero di ricercatori nel gruppo degli s-bot aumenta con un numero più alto di densità di prede per una fissata dimensione del gruppo. Risulta interessante anche la proporzione di raccoglitori nel gruppo di sei o otto s-bot infatti sono uguali quindi questo significa che in media ci sono più ricercatori in quest'ultimo gruppo. Questo fenomeno potrebbe spiegare la perdita di efficienza quando si aumenta la dimensione del gruppo anche quando gli s-bot usano l'adattamento. A causa del particolare set-up che è usato (ad esempio la dimensione dell'arena) non è stato possibile fare esperimenti con un numero maggiore di otto robot.

3.6.3 Selezione del migliore individuo

Il meccanismo di adattamento che si sta studiando si basa solo sui successi o fallimenti dei singoli robot. Se un robot, per un qualsiasi motivo, per quanto riguarda il compito di ricerca è migliore degli altri, avrà più successi e quindi è più probabile che vada a fare parte della categoria dei raccoglitori. Si può arrivare a questa conclusione anche intuitivamente ma è importante anche verificarlo sperimentalmente.

In generale le differenze tra i robot possono essere create artificialmente oppure essere caratteristiche intrinseche agli stessi. Nel primo caso ad esempio è possibile costruire dei robot che siano intenzionalmente designati per il compito di raccoglitori e altri invece essere costruiti con lo scopo di esplorare l'ambiente e riconoscere i punti pericolosi. Nel secondo caso, invece, le differenze vengono dalle imperfezioni dei componenti usati per costruire i robot i quali non potranno mai essere identici gli uni agli altri (ad esempio si possono considerare due motori dei robot che non hanno la stessa potenza). Qualsiasi meccanismo di divisione del lavoro deve tenere conto di questa eterogeneità.

Si mostrerà che l'adattamento individuale può essere efficace per selezionare l'individuo migliore per il task di recupero. Si ricorda che l'algoritmo che qui viene utilizzato innanzitutto non tiene conto della presenza di al-

ID	Tot. Exp	volte raccoglitore
*MindS-bot*1	6	5
*MindS-bot*2	3	2
*MindS-bot*3	9	1
*MindS-bot*4	9	4
*MindS-bot*5	3	0
*MindS-bot*6	10	4

Tabella 3.1: Nella tabella sono riportati gli identificativi dei 6 robot e per ciascuno il numero di esperimenti in cui è stato utilizzato il numero di volte in cui si è riconosciuto nella categoria di raccoglitore

tri robot e che risulta essere diverso dall'originale adattato allo scopo che si vuole raggiungere. Un robot non possiede la conoscenza né di quanti altri robot né di quanti nidi ci siano nell'arena, non esiste un modello specifico dell'ambiente o delle proprie funzionalità all'interno dei robot.

I *MindS-bot* sono costruiti in maniera identica, le uniche differenze provengono dai componenti. Per quanto riguarda gli s-bot, sono state introdotte artificialmente alcune caratteristiche eterogenee. In quanto segue si considerano i robot che possiedono una P_l maggiore a 0,025, modificando la definizione data precedentemente, come raccoglitori. Data la natura stocastica degli esperimenti è possibile modellare il fatto che un robot sia considerato un raccoglitore come un evento casuale.

Robot reali

Sono stati utilizzati i dati provenienti degli esperimenti già citati in precedenza, un totale di 4 robot presi casualmente da un totale di 6. La tabella 3.1 riporta il numero di volte in cui un *MindS-bot* è stato associato alla categoria dei raccoglitori alla fine degli esperimenti.

Simulazione

Sono stati creati sei s-bot differenti i quali possiedono diverse velocità. In particolare la velocità del primo è stata impostata alla metà di quella degli s-bot usati precedentemente, per quanto riguarda la velocità degli altri cinque robot la velocità è stata presa da quella degli esperimenti scorsi scalata di un termine rispettivamente di 0,7 e 0,9. I sei robot sono stati composti combinando tutti i possibili valori formando quindici

gruppi differenti. Ogni gruppo è stato testato con le stesse cinquanta istanze create in maniera casuale con una densità di prede di 0,001 s^{-1}. La simulazione del gruppo è durata 2400 secondi ed è stato contato il numero di volte che ogni s-bot in ogni gruppo ha finito per essere un raccoglitore.

Discussione dei risultati

Se l'adattamento tiene conto delle differenze tra gli individui, la probabilità che ogni robot diventi un raccoglitore alla fine degli esperimenti dipende dagli altri robot dello stesso gruppo o è differente per ognuno dei robot.
L'adattamento non tiene conto delle differenze tra i robot se la probabilità di diventare un raccoglitore è costante per tutti i robot. Per i *MindS-bot* i dati nella tabella 3.1 permettono di confutare l'ipotesi che la probabilità di diventare un raccoglitore è costante con una affidabilità del 95%. Così una delle altre due condizioni può essere ritenuta vera.
Entrambe provano che l'adattamento considera le caratteristiche individuali per applicare la divisione del lavoro. Tramite questo ragionamento si arriva a capire che sia le caratteristiche individuali dei robot sia la capacità media del gruppo sono cruciali per la selezione dei migliori individui. Non sorprende che dai risultati ottenuti si apprende che la probabilità di diventare un raccoglitore è sempre maggiore per i robot che sono più veloci.

3.7 Conclusioni generali

Quando si lavora con un numero di robot alto, il rischio di interferenze tra gli stessi è alto e questo riduce l'efficienza del gruppo. Per risolvere il problema occorre progettare un algoritmo che riesca a fare evitare al robot le zone problematiche, cioè quelle zone in cui si verificano il maggior numero di collisioni tra i robot dovute alle manovre per evitare che gli uni vadano contro gli altri.
Un metodo studiato per risolvere questo problema è quello di dividere l'area totale a disposizione dei robot in settori e poi assegnare ogni settore ad un singolo robot, questo fa aumentare l'efficienza in due fattori: riduce le collisioni tra i robot e rende più veloce la ricerca.

Il modello a soglia variabile risulta essere un algoritmo di divisione del lavoro implementato sui robot e ispirato alla ricerca di cibo delle formiche.

Il modello sottolinea il ruolo che l'apprendimento gioca nei comportamenti collettivi delle formiche durante la ricerca di cibo.

La semplice forma di adattamento, qui esposta, che si ha in ogni singolo individuo e che sfrutta delle informazioni del tutto locali, può migliorare l'efficienza del gruppo.
L'efficienza aumenta attraverso la divisione del lavoro che tiene implicitamente conto delle caratteristiche di eterogeneità del gruppo. I robot non comunicano tra di loro, quindi questo significa che il loro algoritmo di controllo non necessita di sapere quanti individui sono presenti.

3.8 Lavori in fase di studio

Altri lavori collegati a questo, propongono altri metodi per aumentare l'efficienza della divisione del lavoro. Ad esempio nel caso in cui i robot debbano raccogliere oggetti di due tipi differenti, lo studio propone questa soluzione: i robot raccolgono gli oggetti di un tipo o di un altro con una probabilità che cambia seguendo il numero di oggetti presenti nell'ambiente e il numero di robot che hanno come obiettivo quel tipo di oggetto[12].

Un altro studio[13] ha svolto esperimenti con una soglia, affidata ai robot, che viene assegnata all'inizio, e non cambia durante gli esperimenti, in questi si mostra che dei comportamenti complessi in un gruppo, possono essere creati utilizzando anche semplici algoritmi.

Come ultimo un esempio[14] che ha portato a risultati simili a quelli mostrati qui è stato condotto con una serie di robot i quali possiedono una serie di comportamenti associati ad ogni task. Due valori *impatient* e *acquiescence* sono associati a ciascuna serie. Il primo aumenta ogni volta che il robot non svolge un task, fino a raggiungere una particolare soglia. Il robot poi esegue il task e il secondo parametro inizia a crescere. Il robot cessa di svolgere l'attività quando quest'ultimo è troppo alto. I valori delle due costanti vengono modificati per ogni robot seguendo l'esperienza del robot e le informazioni che vengono dal gruppo.

[12] Swarm robotics [Dorigo e shahin 2004]
[13] Krieger e Billeter 2000
[14] Parker 1998 framework L-ALLIANCE

Conclusione

Come si è potuto notare, gli algoritmi ispirati al comportamento delle formiche fanno emergere proprietà interessanti come la flessibilità, la robustezza e il decentramento del controllo. Questi rendono i sistemi ai quali sono applicati in grado di risolvere problemi in cui le condizioni cambiano dinamicamente offrendo così un'elevata tolleranza ai guasti e riuscendo a prevedere il comportamento del sistema.
Inoltre, suggeriscono una strategia per progettare sistemi informatici per il futuro: l'utilizzo di **agenti software**, i quali prendendo spunto dalle caratteristiche di divisione del carico di lavoro sono in grado di modulare i propri comportamenti individuali tramite le modifiche che si verificano, reagendo così in maniera rapida e funzionale.
Gli agenti sarebbero quindi in grado di anticipare collettivamente gli effetti collaterali negativi, dovuti all'evoluzione delle condizioni, o per controbilanciare l'impatto dei pericolosi cambiamenti ambientali.

Dopo aver analizzato le due soluzioni proposte, sono stati confrontati i risultati ottenuti. Vengono ora riprese le caratteristiche fondamentali di ciascuno.

Il primo ad essere utilizzato è stato il metodo a soglia fissa, il quale risulta essere un'ottima base di partenza per lo studio, infatti, si è riusciti a spiegare la flessibilità del comportamento delle formiche, e la loro predisposizione all'adattamento. Per essere veramente sicuri dei risultati ottenuti con le ipotesi fatte, gli esperimenti andrebbero ripetuti con altre specie di formiche. Per quanto riguarda però il livello del lavoro svolto è adatto a quanto si voleva dimostrare con questo documento.

Sicuramente un migliore metodo che mette in risalto le normali differenze tra essere viventi è il metodo a soglia variabile attraverso il quale si è vista proprio la divisione in caste dovuta alla differenziazione dei compiti all'interno del gruppo. Anche se gli esperimenti sono stati svolti con dei robot si è visto che caratteristiche che differiscono da un individuo all'al-

tro sono sempre presenti e queste determinano particolari **abilità** che lo possono inserire all'interno di una o dell'altra casta.

Questo tipo di esperimenti si è sicuramente prestato di più per gli scopi che si volevano raggiungere infatti tramite la facilità d'uso e di costruzione dei robot si è riuscito ad applicare alla realtà gli studi che sono stati fatti tramite la prima metodologia. E' stato possibile implementare un algoritmo ispirato al metodo di sviluppo delle formiche e si è notato, dato che la soglia variava con il numero di successi e insuccessi, che si era in grado di categorizzare gli stessi secondo le loro proprietà.

Certamente a livello di risultati i migliori sono stati introdotti con quest'ultima metodologia, sia perché la parte teorica di simulazione era supportata da una parte pratica (realizzazione robot) sia perché gli algoritmi introdotti sono stati adattati a questa tipologia in modo da avere dei risultati più significativi possibili. E' stato inoltre possibile tramite questo determinare una struttura di robot complessa: cioè i singoli sono a basso profilo mentre si è riusciti a dimostrare che il gruppo si muove secondo precise leggi che oltre alla sopravvivenza della specie rispondono anche all'adattamento. Per adattamento si intende che nei robot si sono fatti strada dei comportamenti più inclini verso determinate operazioni piuttosto che altre.

In conclusione con il perseguimento di studi in campo biologico si è riusciti a concretizzare sviluppi teorici sul ruolo adattativo di organismi viventi, iniziando a conoscere il potenziale di queste forme di intelligenza applicabili alla risoluzione di problemi per noi ancora complessi.

Ringraziamenti

Ringrazio innanzitutto il Professor Andrea Roli per la disponibilità e pazienza mostrata durante tutto il percorso che ha portato dall'elaborazione alla stesura della tesi.

Ringrazio i miei genitori Teresa e Oriano che hanno permesso il conseguimento di questo traguardo, per avermi sostenuto e sopportato nei momenti più bui.

Ringrazio la mia ragazza Alice per aver sempre cercato di capire quali fossero i miei problemi pur sapendo che i suoi sforzi si sarebbero infranti sul muro della mia testardaggine.

Ringrazio mia sorella Laura perché per me è sempre un punto di riferimento e modello.

Ringrazio i miei amici e compagni di corso Angelo, Pietro e Francesca per aver reso molte lezioni più interessanti del normale ed avere condiviso tante belle esperienze insieme; abbiamo imparato a conoscerci sempre meglio sia nell'ambiente universitario sia all'esterno, spero che questa amicizia continui sempre.

Un grazie anche a Francesca, compagna di corso e di laurea per aver condiviso qualche esame con me ma più che mai per quelle volte in cui ci siamo trovati a farci forza a vicenda senza aver mai mollato.

Grazie anche a Marco per la sua disponibilità e pazienza nello spiegarmi le cose che non capivo, amico e anche lui compagno di laurea.

Grazie anche a Luca mio cognato per avermi instradato in questo cammino.

Grazie anche a tutti i miei amici che mi hanno sempre, a loro modo, sostenuto e spronato nel dare sempre il massimo.

In poche parole grazie a tutti quelli che hanno speso, anche per poco, parte del loro tempo aiutandomi in questo cammino, sia per un sorriso sia per un semplice chiarimento.
Grazie a tutti per esserci stati.

Bibliografia

[1] BONABEAU E., THERAULAZ G. Swarm Smarts *Scientific American March 2000, pages 72-79*

[2] BONABEAU E., THERAULAZ G. AND DENEUBOURG J.-L. 1996. Quantitative study of the fixed thereshold model for the regulation of division of labor in insect societies. *Proceeding of the Royal Society of London, Series B-Biological Sciences 263, pages 1565-1569*

[3] BONABEAU E., THERAULAZ G. AND DENEUBOURG J.-L. *Fixed Response Theresholds and Regulation of division of Labor in Insect Societies Bullettin of Mathematical Biology 1998, 60 pages 753-807 Article No. bu980041*

[4] LABELLA T. H., DORIGO M. AND DENEUBOURG J.-L. *Division of Labour in a Group of Robots Inspires by Ants's Foraging Behaviour* Technical Report N TR/IRIDIA/2004-013 October 12, 2006. Published by IRIDIA in ACM Tramsactions on Autonomous and Adaptive Systems, vol. 1, num. 1, pages 4-25

[5] LABELLA T. H. *Prey Retrevial by a Swarm of Robots Technical Report No TR/IRIDIA/2003-16 May 2003 August, 2003*